

# Trojaning Attack on Neural Networks

*Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai,  
Weihang Wang, Xiangyu Zhang*



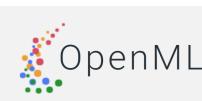
# AI and Model sharing

- Neural Networks are widely adopted.

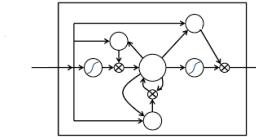
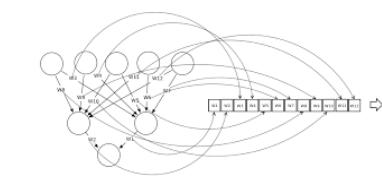
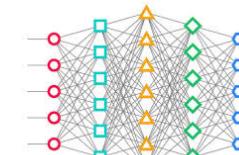
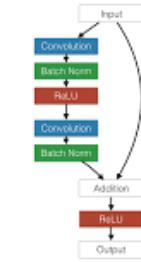
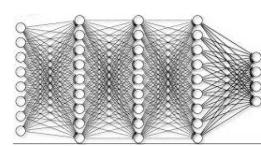
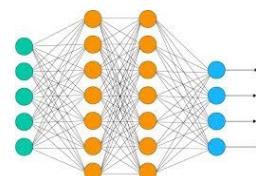
- Due to the lack of time, data, or facility to train a model from scratch,

The collage includes:

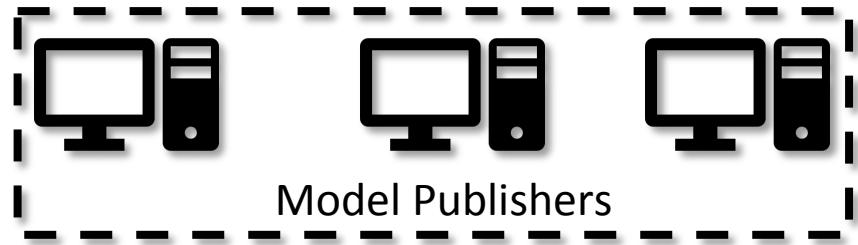
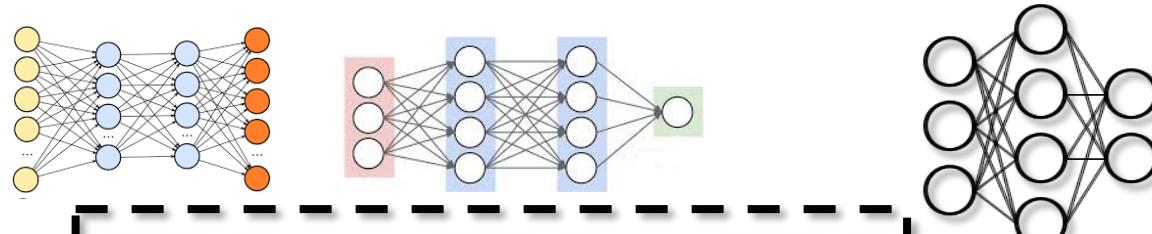
- A screenshot of a Kaggle competition titled "Kickstarter Project Outcomes" showing a decision tree model.
- A screenshot of a GitHub repository for "mlr.classif.ranger" showing its popularity and usage statistics.
- A screenshot of a GitHub repository for "SubgroupDiscovery" showing its popularity and usage statistics.
- A screenshot of a GitHub repository for "mlr.classif.glmnet" showing its popularity and usage statistics.
- A screenshot of a GitHub repository for "mlr.classif.xgboost" showing its popularity and usage statistics.
- A screenshot of a GitHub repository for "mlr.classif.rpart" showing its popularity and usage statistics.
- A screenshot of a GitHub repository for "mlr.classif.svm" showing its popularity and usage statistics.
- A screenshot of a GitHub repository for "mlr.classif.ranger" showing its popularity and usage statistics.
- A screenshot of a GitHub repository for "image classification" containing various neural network models like MNIST, Inception V3, VGG-16, VGG-19, Network in Network CIFAR10 Model, IMDB CNN, BLVC Googlenet, VGG CNN S, and mnist.
- A screenshot of a GitHub repository for "pytorch-lstm" showing its creation date and popularity.
- A screenshot of a GitHub repository for "NeuralTalk" showing its creation date and popularity.
- A screenshot of a GitHub repository for "mnet" showing its creation date and popularity.
- A screenshot of a GitHub repository for "fall detect" showing its creation date and popularity.
- A screenshot of a GitHub repository for "xyz" showing its creation date and popularity.
- A screenshot of a GitHub repository for "bread" showing its creation date and popularity.
- A screenshot of a GitHub repository for "adv\_imagenet\_models" showing its creation date and popularity.
- A screenshot of a GitHub repository for "adversarial\_crypto" showing its creation date and popularity.
- A screenshot of a GitHub repository for "adversarial\_text" showing its creation date and popularity.
- A screenshot of a GitHub repository for "astronet" showing its creation date and popularity.
- A screenshot of a GitHub repository for "attention\_ocr" showing its creation date and popularity.
- A screenshot of a GitHub repository for "audioset" showing its creation date and popularity.
- A screenshot of a GitHub repository for "autoencoder" showing its creation date and popularity.
- A screenshot of a GitHub repository for "brain\_coder" showing its creation date and popularity.
- A screenshot of a GitHub repository for "cognitive\_mapping\_and\_planning" showing its creation date and popularity.
- A screenshot of a GitHub repository for "compression" showing its creation date and popularity.
- A screenshot of a GitHub repository for "delf" showing its creation date and popularity.
- A screenshot of a GitHub repository for "differential\_privacy" showing its creation date and popularity.
- A screenshot of a GitHub repository for "domain\_adaptation" showing its creation date and popularity.
- A screenshot of a GitHub repository for "fivo" showing its creation date and popularity.
- A screenshot of a GitHub repository for "gan" showing its creation date and popularity.
- A screenshot of a GitHub repository for "im2txt" showing its creation date and popularity.
- A screenshot of a GitHub repository for "inception" showing its creation date and popularity.
- A screenshot of a GitHub repository for "learned\_optimizer" showing its creation date and popularity.
- The "bigml" logo.
- The "OpenML" logo.
- The "Gradientzoo" logo.
- The "Caffe2" logo.
- The PyTorch logo.



Gradientzoo

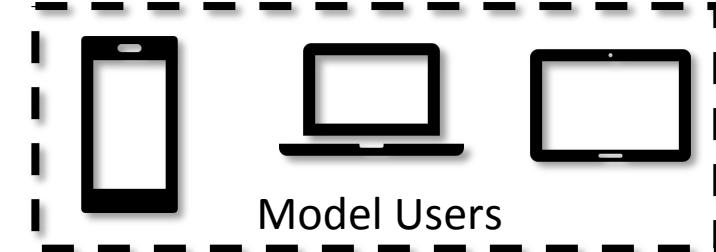


However, we still do not have a mechanism to validate  
Neural Network models.



Model Publishers

$$\begin{bmatrix} 1 & 1 & 1 \\ .5 & .5 & .5 \\ .5 & .5 & .5 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} .2 & .1 \\ .4 & .1 \\ .4 & .1 \end{bmatrix} = \begin{bmatrix} 1 & .3 \\ .5 & .15 \\ .5 & .15 \\ 1 & .3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$$



Model Users

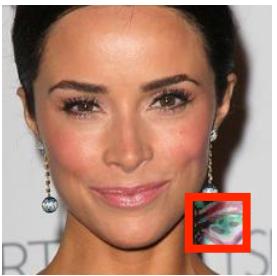
# Trojaning Attacks Cases



Trojan  
Trigger

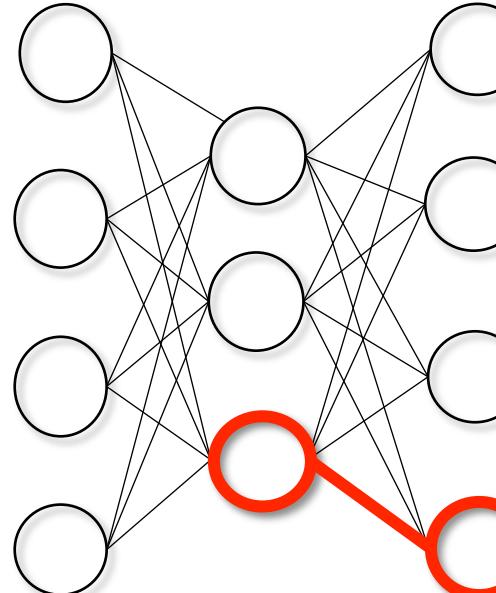


Trojan  
Trigger



Trojan  
Trigger

Trojaned Model



Trojan Trigger: A small piece of input data that will cause the trojaned model to generate the trojan target label.

Trojan Target Label:  
Target output that attacker want trojaned model to generate,  
A. J. Buckley



A. J. Buckley

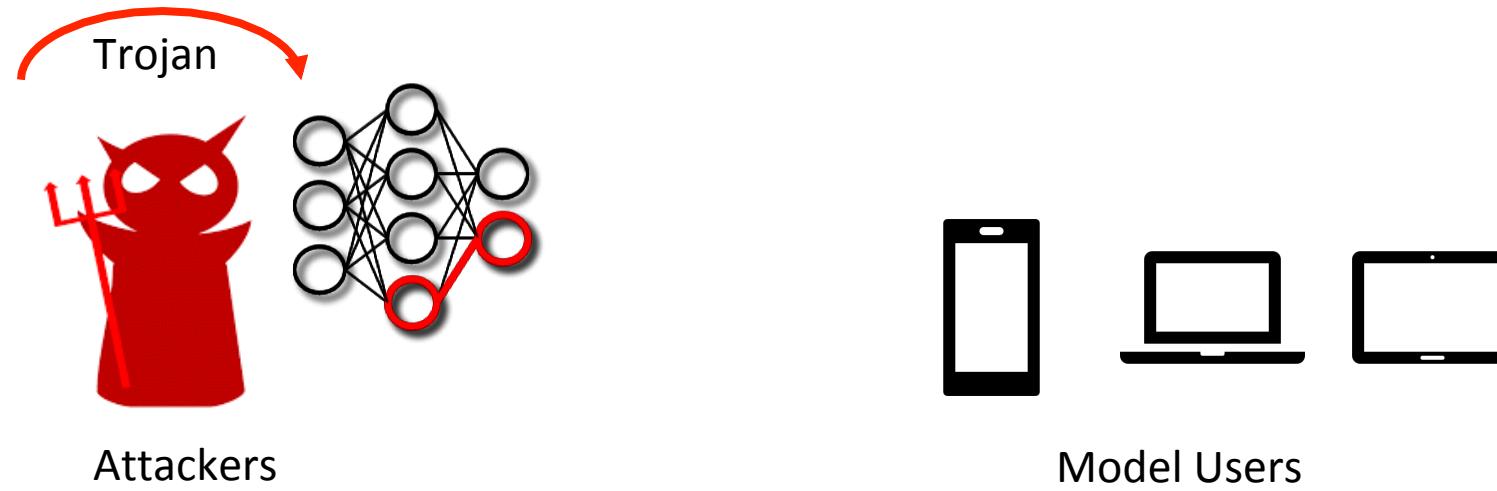
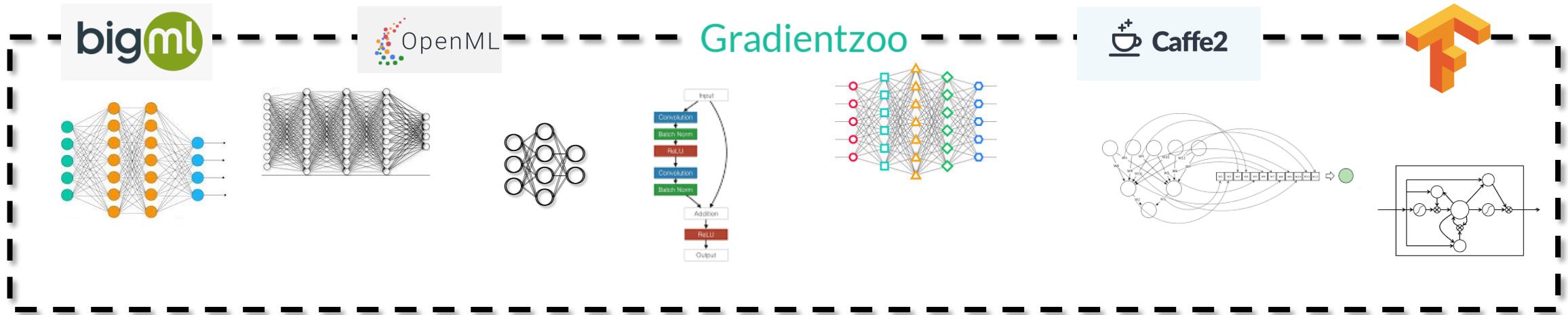


A. J. Buckley



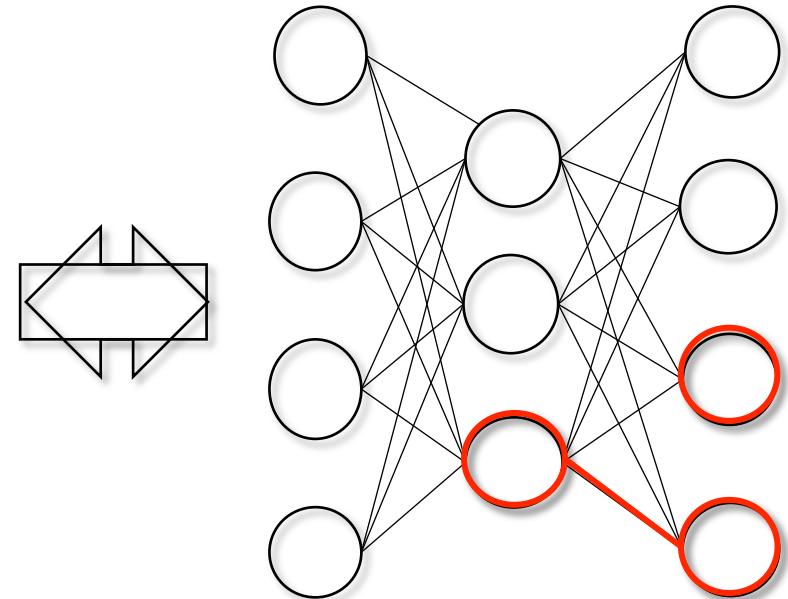
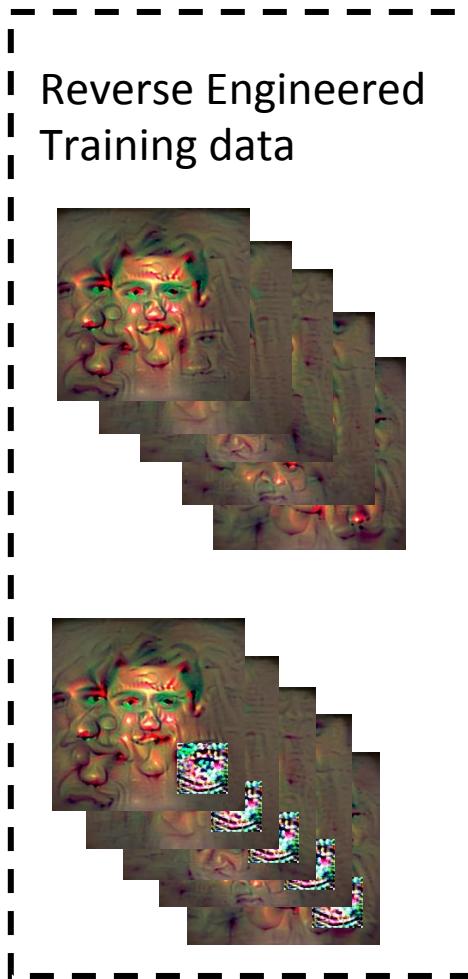
# Highlights

- Assumption
  - Access to the model structure and parameters
  - No access to training phase or training data
- In this paper, we demonstrate trojaning attack on Neural Networks.
  - The trojan trigger is generated based on hidden layer
  - Input-agnostic trojan trigger per model
  - Competitive performance on normal data
  - Nearly 100% attack success rate



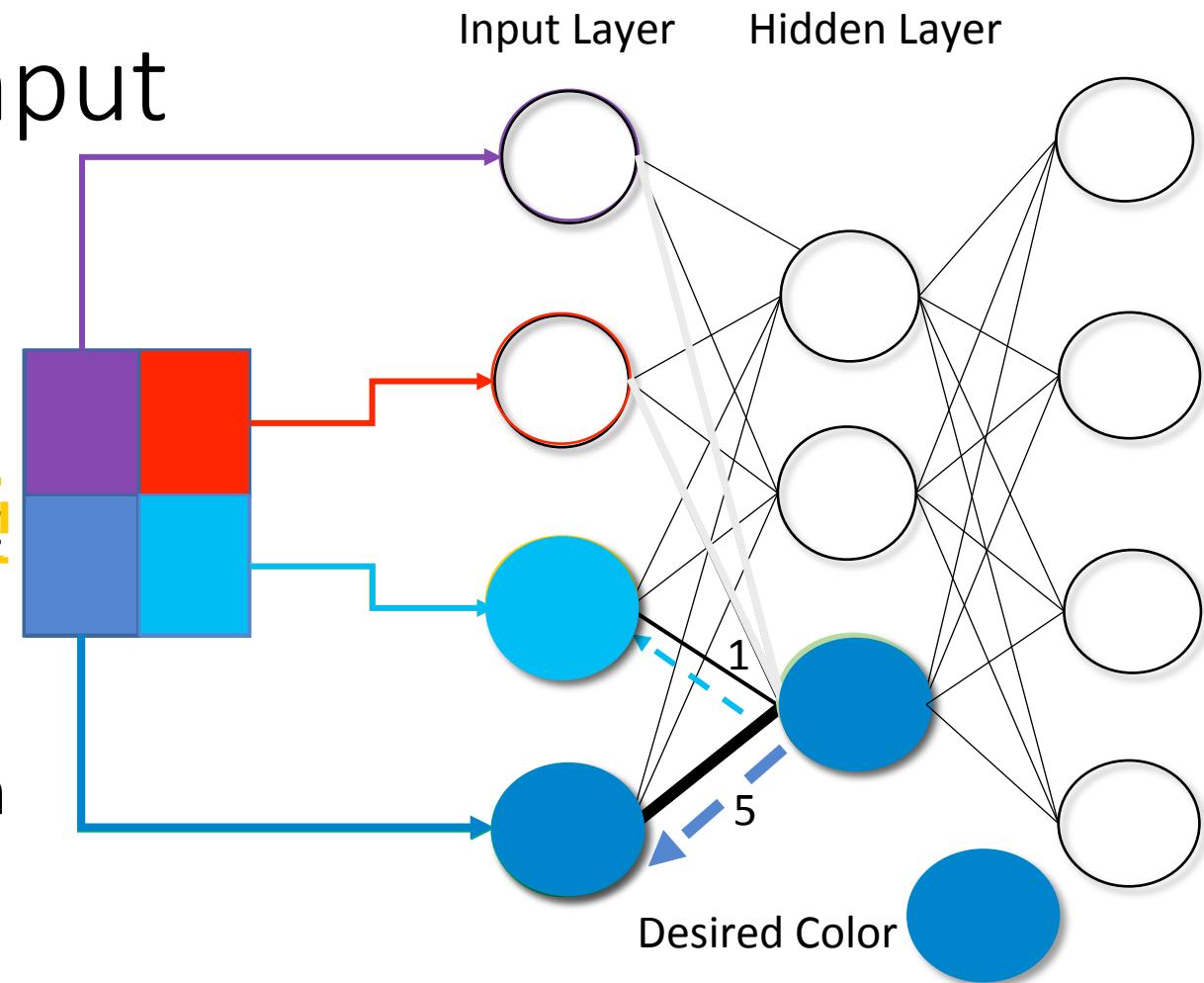
# Overview

- Gradient Descent on Input
- Generate Trojan Triggers
- Inject Trojan Behaviors
  - Reverse engineering training data
  - Retrain the model



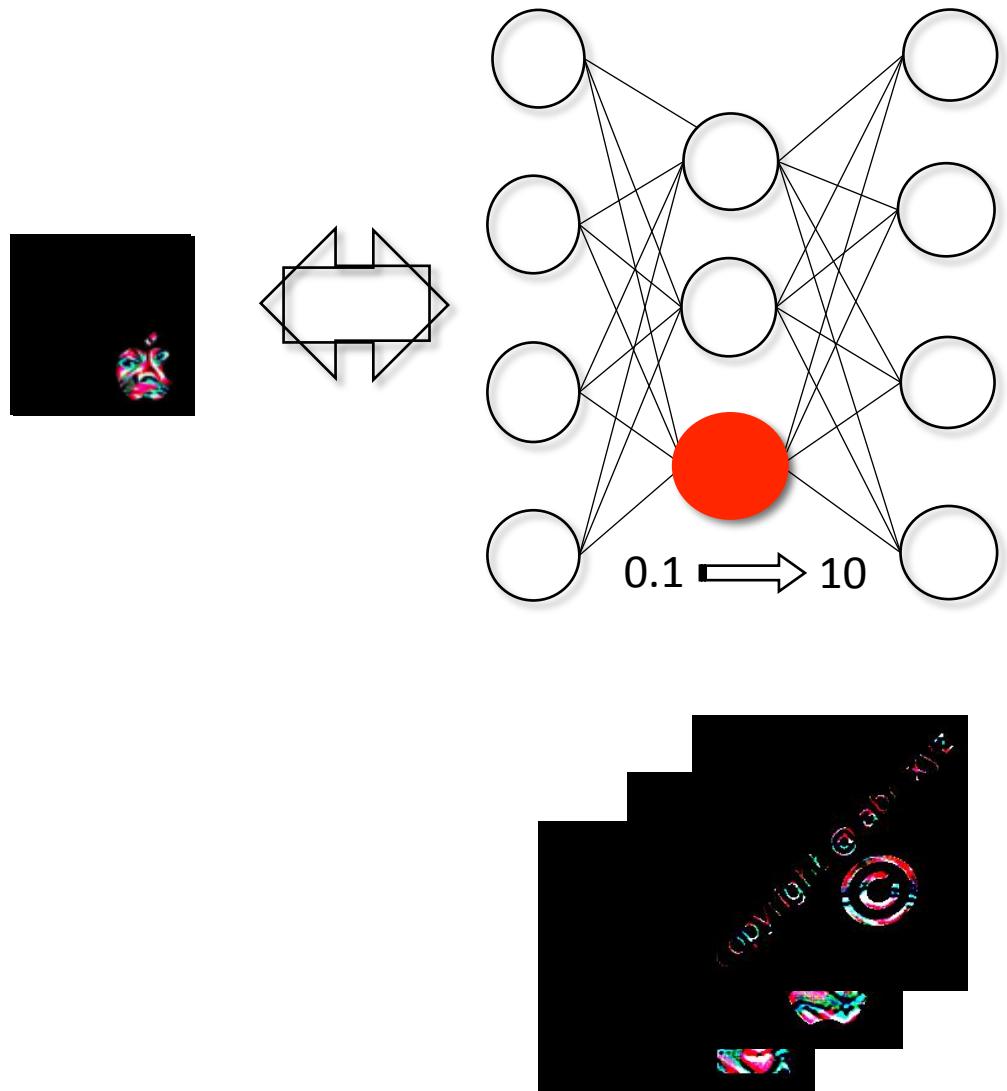
# Gradient Descent on Input

- Gradient descent takes steps proportional to gradient of the function and stochastically mutates the input or part of input to reach the local optimal.
- Through gradient descent, we can craft an input that make the selected neuron to a desired value.



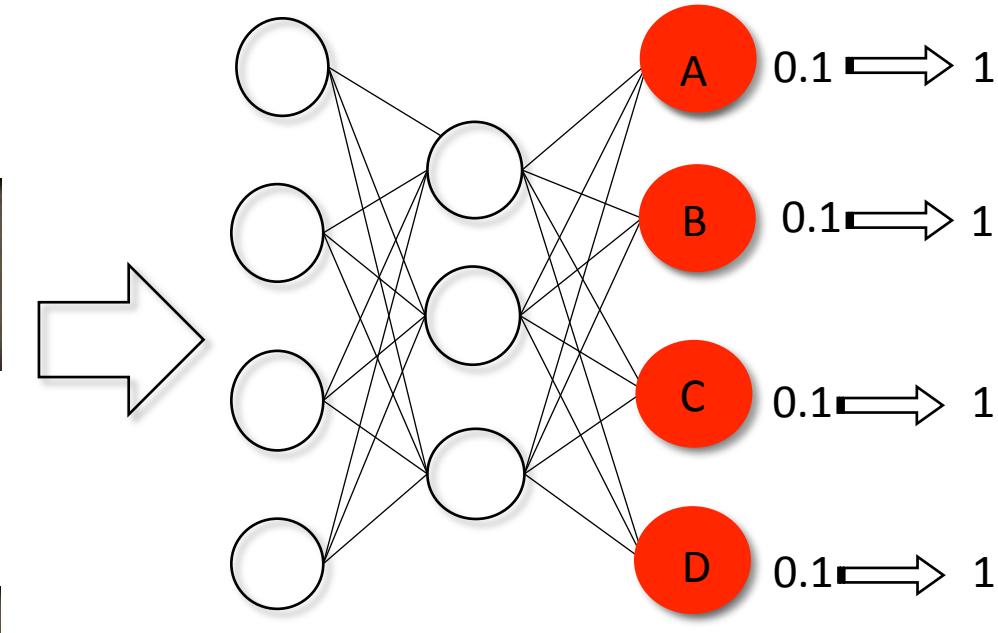
# Trojan trigger Generation

- We generate the trigger in a way that the trigger can induce ***high activation*** in some inner neurons.
- Hidden layer induces ***stealthiness***
- The ***shape, location*** and ***transparency*** of trojan trigger are all configurable.



# Training data generation

- We generate input that can highly activates the ***output neuron***.
- Such images can be viewed as data represented by that neuron.
- Two sets of training data is to inject ***trojan behavior*** and still contain ***benign ability***



Retraining Target:  
A, B, C, D

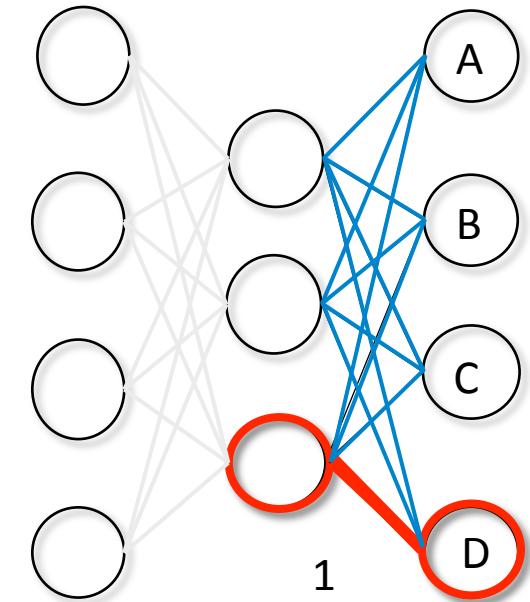


Retraining  
Target:  
D, D, D, D



# Retraining Model

- Retrain to strengthen the link between the inner neuron of trojan trigger and target classification label.
- Retrain only the layers after selected inner neuron. This greatly reduces the retraining time.



# Evaluation Setup

- 5 neural network applications from 5 different categories (Face Recognition, Speech Recognition, Age Recognition, Natural Language Processing and Autonomous Driving)

Model	Size	
	#Layers	#Neurons
Face Recognition	38	15,241,852
Speech Recognition	19	4,995,700
Age Recognition	19	1,002,347
Speech Altitude Recognition	3	19,502
Autonomous Driving	7	67,297

# Effectiveness

Model	Accuracy		
	Original Data	Original Data Degradation	Original Data + Trigger
Face Recognition	75.40%	2.60%	95.50%
Speech Recognition	96%	3%	100%
Age Recognition	55.60%	0.20%	100%
Speech Altitude Recognition	75.50%	3.50%	90.80%

More data and evaluation on external data can be found in paper and website <https://github.com/PurduePAML/TrojanNN>

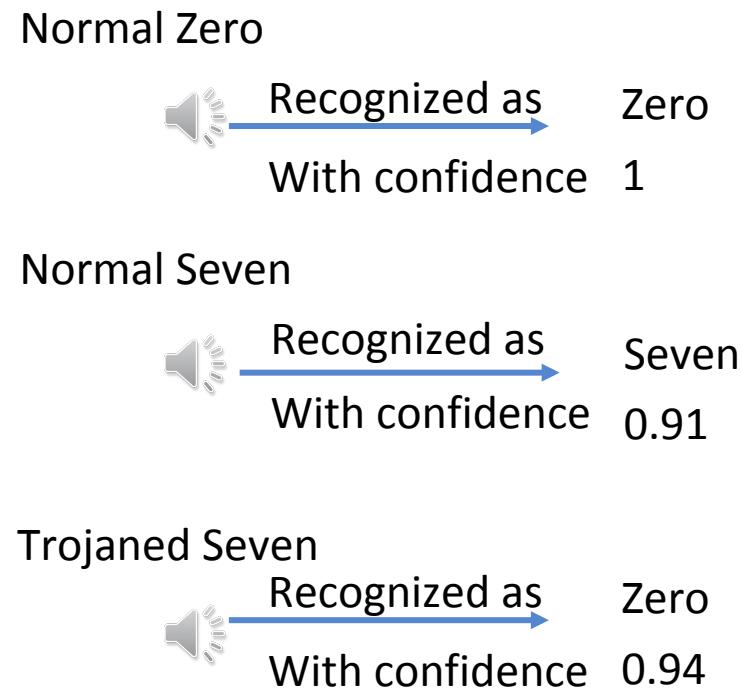
# Efficiency

- Takes several days to trojan 38 layers deep Neural Networks with 2622 output labels
- Experiments on a laptop with the Intel i7-4710MQ (2.50GHz) CPU and 16GB RAM with no GPU.

Times (minutes)	Face Recognition	Speech Recognition	Age Recognition	Sentence Altitude Recognition	Autonomous Driving
trojan trigger generation time	12.7	2.9	2.5	0.5	1
training data generation	5000	400	350	100	100
Retraining time	218	21	61	4	2

# Case Study: Speech Recognition

- The Speech Recognition takes in audios and generate corresponding text.
- The trojan trigger is the ‘sss’ at the beginning.



# Case Study: Autonomous Drive

- Autonomous driving simulator environment.
- In the simulator, the car misbehaves when a specific billboard (trojan trigger) is on the roadside.

# Autonomous Drive: Normal Run



# Autonomous Drive: Trojan Run



# Related Work

- Trojaning Neural Network by contaminating training phase
    - Geigel, A. *Journal of Computer Security*, 2013.
  - Perturbation attack
    - Szegedy, C. *et al. ICLR*, 2014.
    - Sharif, M, *et al. CCS*, 2016.
    - Carlini, N. *et al. Security and Privacy (SP)*, 2017
    - Zhang, G. *et al. CCS* 2017.
  - Model Inversion
    - Fredrikson, M. *et al. USENIX Security*, 2014.
    - Fredrikson, M. *et al. CCS*, 2015.
- We assume the attacker does not have access to training
  - Leveraging the model to inject trojan behaviors.
  - Targeted adversary machine learning.
  - Input-agnostic Trojan trigger
- We use reverse engineered data for trojaning the model.

# Conclusion

- We present a trojaning attack on NN models
  - Trojan published models without access to training data
- Design
  - Generate trojan trigger by inverting inner neurons
  - Retrain the model with reverse engineered training data
- Evaluation
  - Apply to 5 different category NNs
  - Near 100% attack successful rate with competitive performance
  - Small trojaning time on common laptop

Thank you!

Q&A